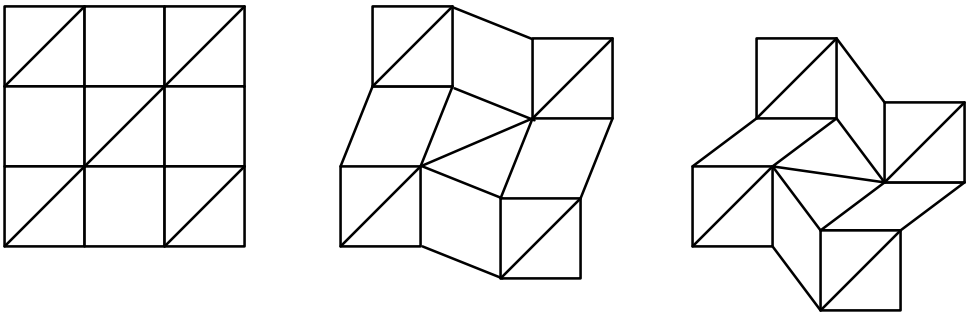
FLOPPY GRIDS - Discovering the Mathematics of Grid Bracing

A COMAP FAIM Project by G. Klatt, CC Edwards, S. Heubach, and V. Howe



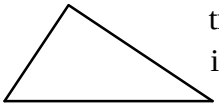
ABSTRACT

We examine the problem of determining when a rectangular grid of squares with some diagonal braces is rigid. We use this problem to model the process of making and testing conjectures, and to show a mathematical approach to problem solving. Through the modeling process, we recast the problem into a bipartite graph version and a matrix version to illustrate the advantage of translating a problem into a different setting and to motivate a brief look at those areas of mathematics. A highlight for students is a game version of the grid bracing problem that encourages their discovery of good tests for grid rigidity.

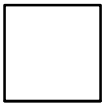
TABLE OF CONTENTS

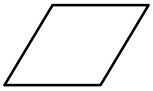
SECTION 1 INTRODUCTION

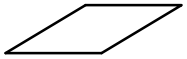
You probably know that a triangle is inherently rigid. That is, if you build a triangle with three sticks attached at the ends with nails, it won't deform its shape. In contrast, a square built the same way can deform into a variety of parallelograms, or even fall completely down.



triangle
is rigid







square
is floppy

1.1 STATEMENT OF THE PROBLEM

The problem we will examine is how to make a grid of squares rigid by adding diagonals. If we add all the diagonals, the resulting grid is rigid because it is made up of triangles. Can we do it with fewer diagonals? For example, we show a 2 by 2 grid with 3 diagonals. Is it rigid?

RIGID!

RIGID?

ACTIVITY 1*

i) Decide if a 2 by 2 grid with 3 diagonals is rigid. If you decide it is not rigid, show how it can be deformed. If you decide it is rigid, give an argument why that is so.

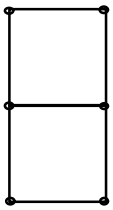
ii) Does it matter which way the diagonals go?

iii) Does it matter which of the four diagonals is missing?

At this point you may want to build a physical model to help investigate or to convince someone you are right. See exercise 1.4 for one way to do that.

* Note: Answers and comments on the activities are given in Appendix I. You will get the most value by trying the activities first on your own (or in groups). Refer to the answer/comments only if you can't get a satisfactory result on your own.

1.2 SOME ASSUMPTIONS AND OBJECTIVES



- We are assuming plane figures. That is, they must stay in the plane of the paper.
- We will use the name "grid" to mean a rectangular array of squares with some (or all or none) having diagonal braces. These grids are modeling a physical array where the sides are stiff (boards or metal rods) and there is a connector at each corner that will allow these rods to rotate. The braces are also stiff rods pinned at the corners. In particular, a 2 by 1 grid with no braces is made of 7 rods (edges) and 6 flexible connectors.
- Later it will be very convenient to have the number of rows less than or equal to the number of columns. For example, we will use 2 by 3 grids rather than 3 by 2 grids. For all of the questions we will examine (like grid rigidity) this assumption will not cause any loss of generality.

Here are some comments on what we will be doing and why.

- Although grid bracing has practical applications, our primary interest is in the challenge of problem solving.

•We will try to discover "good" methods of deciding if a given grid braced with some diagonals is rigid or not.

•We will introduce "the bracing game" in two versions. In the "building" version players take turns adding braces trying to make the grid rigid. In the "gutting" version we start with braces in every square; the winner is the first to make the grid floppy (not rigid).

•Beyond the immediate problem of deciding whether a given grid is rigid we will be concerned with illustrating how a mathematician goes about tackling a problem. We will bring up ideas of necessary and sufficient conditions, how to make and test conjectures, the role of lemmas, theorems, corollaries and algorithms, and the advantage of translating a problem from one structure into another.

•Along the way you will see applications of graphs and trees and of matrices to our problem of grid bracing.

EXERCISES FOR SECTION 1

1.1 How many diagonals does it take to make a 1 by 2 grid rigid? ...a 1 by 3 grid? ... a 1 by n grid?

1.2 Show how a 2 by 2 grid with any arrangement of just 2 diagonals can be deformed. For example, show  is not rigid.

1.3 Consider making a pentagon rigid with diagonals. We claim any two diagonals will make a pentagon rigid, and with just one diagonal it will be floppy.
How about the same question for hexagons? Can it be done with 2? ...with 3? Will any choice of 3 diagonals make it rigid? (See section 7.2 for a little more on this.)

1.4 A physical model helps a great deal in deciding rigidity. Use your ingenuity and the materials at hand to build a model. Here is one way to do this with strips of cardboard and thumbtacks. Cut 1/2" strips from the cardboard backing off a pad of paper. Make the cuts parallel to the long side so you have a bunch of 1/2" by 11" strips. Cut some in thirds and make thumbtack holes exactly 3" apart. Cut some in half and make thumbtack holes exactly 4 1/4 " apart. Make a 2 by 2 grid with the short strips with (flat head) thumbtacks holding the corners. Add diagonals to three of the squares with the long strips. Is it rigid? How many short and how many long pieces are needed for putting all diagonal braces in the 3 by 3 grid? DANGER! WARNING! Do not leave this sitting around with the tacks pointing up. Someone could get hurt badly if they leaned their hand on it. For safety you should always work with the points down.

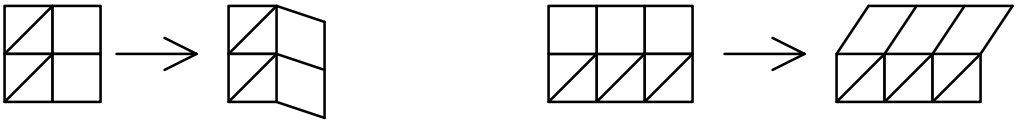
2 FIRST PRINCIPLES OF GRID BRACING

2.1 A PARTIAL CONDITION

Here is a partial condition for rigidity in any grid.

The Empty Row or Column Principle (EROC)
If a row or column has no braces (i.e. is empty), then the grid is floppy.

We illustrate how to deform some examples. For now we will be content to motivate this principle with examples. In later sections we will be able to give a more formal proof.



2.2 A LITTLE LOGIC

We would like to pause here to explore some logical consequences of the EROC Principle. We also want to illustrate correct use of the words "necessary" and "sufficient" and the ideas of contrapositive and converse of an implication.

The context for this discussion is having an **implication**, a statement of the form if (something) then (something else). Rather than talk in general let us use a specific implication: if a number ends in 0 then it is even (more formally, if n is a multiple of 10, then n is a multiple of 2). This is a true implication: if n = 10m, then n = 2(5m).

Many people carelessly think every implication can be reversed. Is the following a true statement?
If n is a multiple of 2 then it is a multiple of 10. __ true __ false
The reverse of an implication is called its **converse**. Here the converse is false. You can show it is false by coming up with a **counterexample**, a number like 8 which is a multiple of 2 but not of 10. In general a counterexample to an implication is an example which makes the "if" part true and the "then" part false.

Here is another variant of the original implication. It is called the **contrapositive**, and we form it by reversing and negating the if and then parts. See if you think it is true.
If n is not a multiple of 2, then n is not a multiple of 10. __ true __ false
This turns out to be true. Actually, it is a basic principle of logic that the contrapositive is logically equivalent to the implication - they are really saying the same thing.

Another way of phrasing an implication uses necessary/sufficient language. This is easy to mess up unless you think carefully. Which of the following is true?
For n to be a multiple of 10 it is necessary that it be a multiple of 2. ___
For n to be a multiple of 10 it is sufficient that it be a multiple of 2. ___
The first is true, the second is false. Again n = 8 shows the second is false.

ACTIVITY 2

i) Find a grid with no empty rows or columns which is not rigid. Can it be done with a 2 by 2? ... with a 2 by 3? with a 3 by 3?

ii) Decide which of the following statements are true by EROC, and which are false by part i).

- If a grid is floppy then it has an empty row or column.
- If a grid is rigid then it has no empty row or column.
- If a grid has no empty row or column then it is rigid.

iii) Decide which of the following statements are correct.

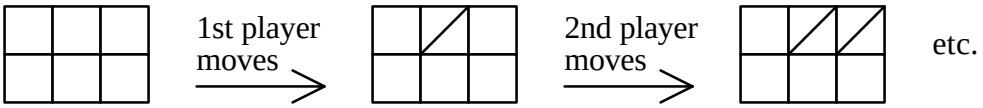
- For a grid to be rigid it is necessary that no row or column be empty.
- For a grid to be rigid it is sufficient that no row or column be empty.
- For a grid to be floppy it is necessary that a row or column be empty.
- For a grid to be floppy it is sufficient that a row or column be empty.

2.3 THE BRACING GAME

We would like you to discover more principles of rigidity like EROC. The ultimate objective is to find an easily tested principle that is both necessary and sufficient. To help you in this quest we introduce a game which requires some skill at deciding rigidity.

THE BRACING GAME - BUILDING VERSION

Start with a given grid size and no diagonal braces. For example, a 2 by 3 empty grid. Players will take turns adding a brace. The winner is the first to make the grid rigid. Part of winning is to support your claim that the grid is rigid with a convincing argument.



THE BRACING GAME - GUTTING VERSION

Start with a given grid size and all the squares braced. Players take turns removing a brace. The winner is the first to make the grid floppy (again with a convincing argument).

For both versions and a given grid size, the first one to move has either an advantage or a disadvantage. Agree who will go first or flip a coin to decide who will go first.

ACTIVITY 3

i) Play the bracing game (building version) at least 6 times with a partner. Perhaps play 2 times with a 2 by 3 grid, 2 times with a 3 by 3 grid and 2 times with a 3 by 4 grid. Take turns playing first. Don't worry too much about winning, but try to come up with good strategies of play and how to tell when the grid is rigid or when it is floppy. Repeat with 6 games in the gutting version. Remember you haven't won unless you can demonstrate that the grid you leave is rigid (building version) or floppy (gutting version).

ii) In round 2 you and your partner will play as a team against another team. Prior to playing round 2, share your good ideas with your partner. You will understand them better if you explain them, and your partner may help you improve them. Now play round 2 with the same size grids. The real winners are the partnerships that learn how to play the game effectively.

iii)Decide if the first player can always win in the 2 by 3 game if they play well. Do the same for the 3 by 3 game. (See section 7.4 for a project on analysis of this game.)

2.4 MORE PRINCIPLES

Here are two examples of proposed principles which may help decide if a given grid is rigid or floppy. The first one is good (correct, useful), the second one needs some work.

THE BUILDER'S LEMMA

For any 2 by 2 subgrid with 3 of the 4 squares braced, the fourth square will act like it is also braced. So we can add its brace without changing the rigidity of the grid.

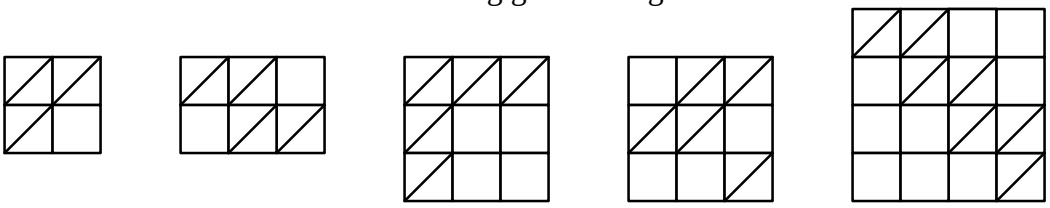
This is based on the result of Activity 1 where we found that a 2 by 2 grid with 3 bracing diagonals is rigid. (The argument is all four angles in each of the three braced squares must be right angles. At the middle corner the three right angles leave no choice but a right angle for the unbraced square. That one right angle forces the other three angles also to be right angles.) Since we have a supporting argument, we have moved up from the status of a **conjecture** (inductive: based on guessing a general result from examples) to a **theorem** (deductive: proved true for all cases). We call it a lemma instead of a theorem because it is not terribly interesting or useful in itself, but is used to establish another result. Here is how we can use the Builder's Lemma to decide rigidity.

THE BUILDER'S ALGORITHM

For each 2 by 2 subgrid with three braces, fill in the fourth (maybe with a dashed line). Keep doing this using original and dashed braces until all squares are filled in or you can't proceed. We can conclude from the Builder's Lemma that if you have filled in all the squares then the grid is rigid. We may conjecture (guess) that the converse is true as well - if you don't have them all filled in, then the grid is not rigid. For now we will only be safe in using the one direction of the implication.

Thus we have the following sufficient condition: **for a grid to be rigid it is sufficient to be able to fill in all the diagonals with the Builder's Lemma.**

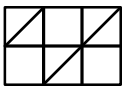
Here is another conjecture that some students came up with in Activity 3. It attempts to capture the observation that all of the following grids are rigid.



THE SIDE-TO-SIDE CONJECTURE

When there are enough diagonals to go from one side to the other, the grid is rigid.

As it stands the side-to-side conjecture is too vague. What is meant by "to go from"? Does the 2 by 3 shown to the right allow you to go from right side to left side? This one is not rigid, so the answer had better be "no".



ACTIVITY 4

i) Apply the Builder's Algorithm to each of the five grids pictured above - are they all rigid?

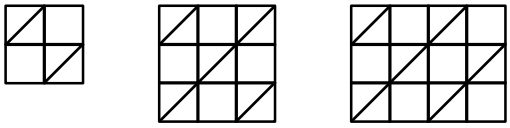
ii) Try to capture the idea of the side-to-side algorithm and express it clearly. Test it with a variety of examples to see if it really works. Clarify if it is a necessary condition, a sufficient condition, both or neither (as a condition for rigidity).

ii) Take your favorite observation about how to tell if a grid is rigid or floppy and try to write it down carefully and clearly. Decide if it is a necessary condition, a sufficient condition, both or neither (as a condition for rigidity).

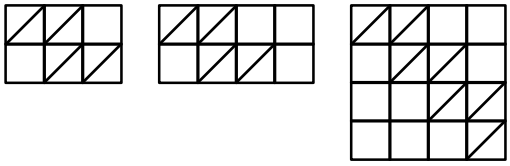
2.5 SOME USEFUL CLASSES OF GRIDS

Here are some grid types we will have use for now and in future sections.

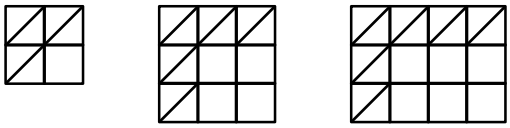
The **checkerboard grids**
start by bracing the top left square, then brace every other square in each row and column



The **staircase grids**
start by bracing the top left square, then repeat bracing right, bracing down until you have to stop



The **gamma grids**
brace the entire first row and first column (called gamma grids because they look like the Greek letter gamma, Γ



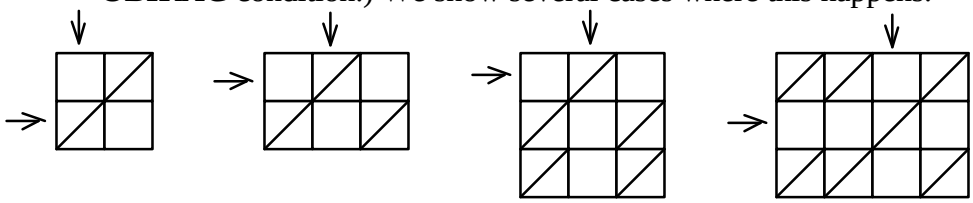
In the next section we will give you some more ways of deciding if a given grid is rigid which were thought up by some clever people who worked on this problem.

EXERCISES FOR SECTION 2

2.1 Apply the Builder's Algorithm to each of the grids shown in Section 2.5 above. What do you conclude in each case? Generalize from this and EROC to conclude the staircase grids are rigid if and only if (give a condition on m and n). Will all the gamma grids be rigid?

2.2 On page 1 there is a picture of the 3 by 3 checkerboard grid deforming - it is not rigid (contrary to my intuition at first). Decide if other checkerboard grids can be deformed in a similar way. Are all of them floppy?

2.3 Here is another candidate for a partial rigidity condition. If a grid has a braced square which is the **O**nly **B**race **I**n its **R**ow **A**nd its **C**olumn, then the grid is floppy. (Call this the **OBIRAC** condition.) We show several cases where this happens.



i) Show how to deform each of these grids.

ii) Show OBIRAC is only a sufficient condition for floppiness by finding a floppy grid where there are no cases of OBIRAC. Hint: look in section 2.5.

iii) As stated OBIRAC is in "if ..., then ..." form. Restate it in its contrapositive form.

2.4 The gamma grids show it is possible to make an m by n grid size rigid with m+n-1 braces. We claim this is a "**Magic Number**": no grid can be rigid with fewer than m+n-1 braces.

i) Show there is no way to make a 2 by 3 grid rigid with just 3 braces.

ii) Restate this Magic Number condition in if _____ then _____ form.

iii) Having at least m+n-1 braces is a necessary condition for rigidity. Show with an example that this counting condition is not sufficient.

2.5 How many different 2 by 3 grids are possible? Here m+n-1 = 4. How many have 4 braces? How many with 4 braces are rigid? What is the probability that if you brace 4 squares at random the grid will be rigid? Same questions for 3 braces. ...5 braces. Try the same questions for the 3 by 3 grid if you feel ambitious. See section 7.6 if you like counting questions of this kind.

2.6 Try to find a grid which is rigid, but where the Builder's Algorithm fails to fill in all the squares. (We will revisit this task in the next section - by then we will have better tools.)

2.7 Assume you want to build a bigger structure from pre-fabricated smaller grids, i.e. they all have braces in the same positions. Since you saw the builder's algorithm, you may suspect that you could mix rigid small grids with non-braced grids but still attain rigidity of the larger structure. For example, if you start with four 2 by 3 grids, can you brace the 2 by 3 grids in such a way (with 4 braces each) that you can put three braced grids and an unbraced 2 by 3 grid together to form a rigid 4 by 6 grid? Try out a few of the rigid bracings that you have already seen. Will every such bracing work? Will none work? Is there at least one (describe it).

2.8 In the above exercise, we started with rigid 2 by 3 and added a floppy 2 by 3. We can also turn the question around and try to use floppy 2 by 3s to make a rigid 4 by 6? Is this possible? First determine how many braces you need at least in the 2 by 3 to make the bigger grid rigid. Then try to find a bracing for the 2 by 3s that will achieve the task or conclude it is impossible. You are allowed to flip the grids over horizontally or vertically before connecting them for the bigger grid.

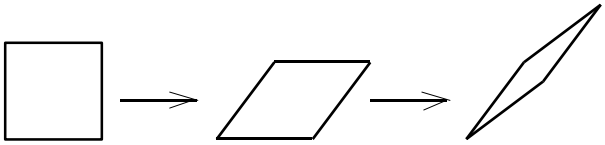
SECTION 3 ALGORITHMS FOR DECIDING RIGIDITY

In this section we will explain two good ways of deciding the question of whether a given grid is rigid or floppy. Both are based in geometry. The first exploits the idea that in deforming a single square (a 1 by 1 grid) the sides will stay parallel. The second improves the Builder's Lemma into a necessary and sufficient condition for rigidity.

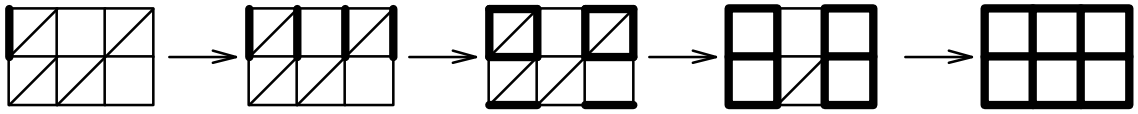
3.1 WALT'S PERSISTENT PARALLELS ALGORITHM

We wish to acknowledge a debt of gratitude to Prof. Walter Meyer who suggested the problem of grid rigidity to us and who came up with this very nice method. It is based on the following geometric observation.

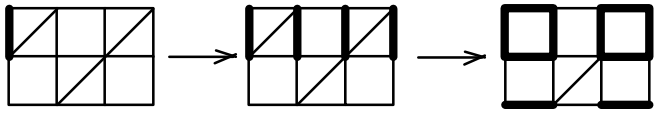
THE PERSISTENT PARALLELS LEMMA
When a square deforms, the sides stay parallel.



Walt's method (which we will call Persistent Parallels or PP for short) starts by picking one edge (say the top left vertical edge) to nail down. Then we proceed to mark all the edges that must stay vertical and the edges that must stay horizontal. A diagonal brace allows us to move from a must-stay-vertical to a must-stay-horizontal (and vice versa). Here is Persistent Parallels in action. The top example is rigid by Builders and the bottom example is floppy by Exercise 2.5.



start	these stay vertical	use 1,1 and 1,3 braces to shift to horizontals	use 2,1 brace to shift to vert. edges in 2nd row	use 2,2 brace to shift to horiz. in 2nd column: RIGID!



can't proceed: FLOPPY!

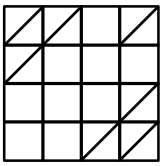
ACTIVITY 5
Using the same 2 by 3 grids as shown above, investigate whether the idea will work if you start by nailing down some other first edge. For example, nail down the bottom left horizontal to start. Then mark those other edges that must stay horizontal and vertical as we did above. PP is working if it concludes the first grid is rigid and the second isn't. Try a variety of different starting edges. Try nailing down some diagonal brace: does PP still work?

3.2 THE BETTER BUILDER'S ALGORITHM

We saw the Builder's Algorithm back in section 2.4. Any 2 by 2 subgrid with 3 braces allows us to fill in the fourth brace. To check rigidity we use the Builder's Lemma repeatedly to fill in braces where we can. If we end up with all squares filled in with braces, then the grid is rigid. However, as you may have seen in Exercise 2.6, there are examples where the grid is rigid even though we couldn't fill in all the squares with the Builder's Algorithm (you will see one such example in Activity 6 below). Therefore, the ability to fill in all the squares with the Builder's Algorithm is a sufficient but not necessary condition for rigidity.

ACTIVITY 6

We show a 4 by 4 grid with 7 braces to the right. Note by the Magic # Condition, 7 is the minimum number of braces needed here to get a rigid grid. (The Magic # Condition was mentioned in Exercise 2.4.)



- See how far you can get with the Builder's Algorithm.
- Apply the Persistent Parallels algorithm. Is it rigid?

We need to improve the Builder's Algorithm so it will work for all grids. The difficulty with the 4 by 4 example is that the braces are "too far apart" for our 2 by 2 Builder's Lemma. Here is the fix that will make it work.

BETTER BUILDER'S ALGORITHM

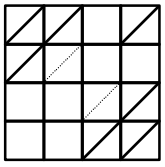
Step 1 Start with the Builder's Algorithm - that is, fill in the fourth square in any (adjacent) 2 by 2 with 3 squares braced.

Step 2 If you get stuck, look for any cases of "non-adjacent" 2 by 2's with 3 braces, and fill in the fourth.

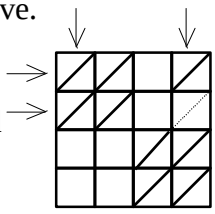
Now repeat Steps 1 and 2 until all squares are braced or there are no more cases of adjacent or non-adjacent 2 by 2's with 3 braces. In the first case (all braced) the grid is rigid; in the second it isn't.

Here is how Better Builder's works on our pesky 4 by 4 from Activity 6 above.

Builder's gets us only this far:



We see a non-adjacent 2 by 2 with 3 braces in the first and second rows and the first and fourth columns. So we fill in a brace in 2,4.



Now use Builders (adjacent 2 by 2's) to fill in all the rest of the squares.

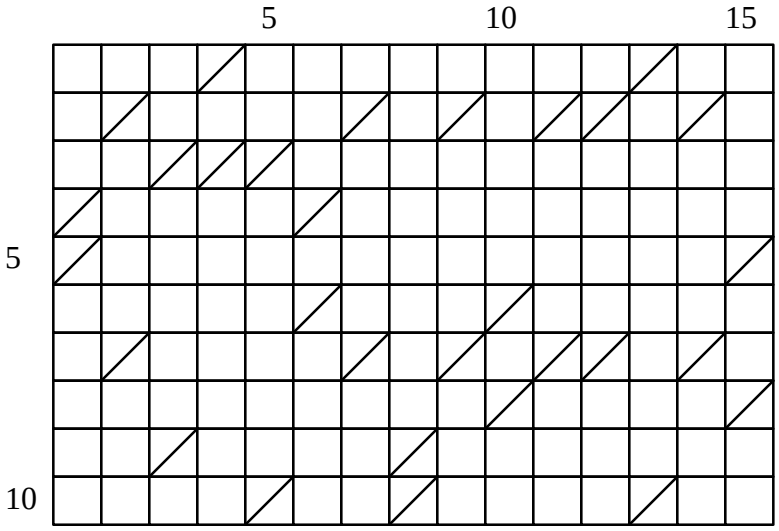
We claim Better Builders will work to decide rigidity for any grid. One argument is fairly easy to give in the graph context of section 6. The crux of the argument is justifying the bracing of the fourth square for non-adjacent 2 by 2's. In Exercise 3.3 we ask you to use an argument based on Walt's Persistent Parallels Lemma for this.

3.3 BRIEF SUMMARY OF WHAT WE KNOW*

At this point we have two algorithms for deciding rigidity, and several partial conditions:
- PP: the grid is rigid if and only if the Persistent Parallels algorithm marks all the edges
- BB: the grid is rigid if and only if the Better Builders algorithm fills in all the braces
- EROC: if a grid has an Empty Row Or Column then it is floppy
- OBIRAC: a grid is floppy if it has a brace which is the Only Brace In its Row And Column
- MAGIC NUMBER: if there are fewer than m+n-1 braces then the grid is floppy

*Mathematicians are careful to distinguish what is really known (i. e. what has been proven) from what is only a good guess based on experience (a conjecture). In this spirit we note the proof of BB has been an exercise; OBIRAC and MAGIC NUMBER will be proved in section 5.

Shown below is a 10 by 15 grid with 30 braces. We call it **Monster Grid #1.** Right now we have only the Persistent Parallels (PP) and the Better Builder's (BB) algorithms for the formidable task of deciding rigidity. In the next sections we show how to (maybe) quickly see rigidity of such a large grid, and how to ask a computer or calculator for help.



MONSTER

GRID #1

10 BY 15

(m + n - 1 = 24)

30 braces

EXERCISES FOR SECTION 3

3.1a) Apply Walt's Persistent Parallels algorithm to the 3 by 3, 3 by 4, and 4 by 4 checkerboard grids, the same size gamma grids, and the same size staircase grids.

b) Show the Better Builders algorithm cannot add any braces to these three checkerboard grids. (So it concludes they are floppy.) Does Better Builders behave any differently than Builders did on gamma or staircase grids?

3.2 Take the 4 by 5 checkerboard grid and move any one brace to an empty square. Use either PP or BB (your choice) to decide if the result is rigid or floppy.

3.3 Use the idea of Walt's PP algorithm to justify adding the fourth brace when the 2 by 2 subgrid does not involve adjacent rows and columns.

3.4 We claim the 3 by 3 checkerboard can be seen to be floppy another way besides OBIRAC. Note that 5 is the magic number for a 3 by 3 (m+n-1=5). Looking at the four corner braces we can conclude that any one of them is **redundant** - if it weren't there, we could fill it in by better builders. Thus we can take out a corner brace leaving only 4 braces, less than the Magic Number required for rigidity. Develop another algorithm that identifies and erases redundant braces (that rhymes!). Make up a good name and see if your algorithm really works (can you erase *any* of the four corners?).

3.5 Decide if Monster Grid #1 is rigid by applying both PP and BB. . Start with your favorite algorithm and compare them for speed and reliability. Note that there are no instances of EROC or OBIRAC which would make things easy.

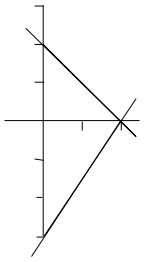
3.6 Find a 3 by 4 grid with 6 braces for which Builders fails to fill in all the braces, but which is rigid. Check that Better Builders will fill in all the braces.

3.7 Modify both versions of the Building Game to start with some random collection of braces. For example, with a 6 by 6 grid size throw a pair of red and white dice a bunch of times to put in the random braces (red die gives row, white die gives column). Maybe throw the dice once to determine how many random braces to add, and each of you throw the dice to see who goes first. Or make up rules for your own version of the game. The objective is to see quickly when a grid is rigid and how to make it so or avoid making it so.

SECTION 4 OTHER SETTINGS FOR THE RIGIDITY PROBLEM

When we do mathematics we often transfer the problem we are working on into a different setting. For example, instead of working with two lines A and B (geometric objects) we replace them by two linear equations (algebraic objects). Then if we want to know if lines A and B intersect (a geometric concept) we can just solve a system of two linear equations (an algebraic concept).

You have done this so often with lines you almost don't distinguish in your mind between the equations $3x - 2y = 6$ and $x + y = 2$ and the two actual lines. When you are asked if these two lines intersect you probably would just use algebra to solve the system of two equations. What you have done is replace geometry by algebra and thus are able to use the tools you have learned in algebra to solve a geometry problem. This is one of the most powerful ideas of mathematics. In this section you will see how to recast the Grid Rigidity Problem into several other settings.

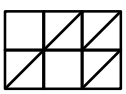


4.1 TABLES AND THE GRID MATRIX

If you have been faithfully doing the activities up to this point you may have noticed that it can be tedious applying the rigidity algorithms like BB or PP even to small grids, and downright difficult for the 10 by 15 Monster Grid. It would be nice if we could get the computer to help us with the work of deciding rigidity.

Take a moment to consider how you would communicate a grid problem to the computer. How do you tell the computer what the grid looks like and where the braces are? Your eyes are very powerful and notice changes in the grid (i.e., when braces are added or deleted or the pattern of braces) almost without thinking about it. What you have trouble with is deciding quickly if a grid is rigid. So you automatically know where the braces are but have trouble processing long and tedious calculations quickly and accurately. The computer has just the opposite problem. The computer has no spacial or intuitive understanding of the actual grid and bracings but, if it has the information in the proper fashion, it can process the information very fast. Our problem is to find an efficient way to communicate the pattern of the grid to the computer in a way the computer can understand.

You probably have already thought of a good solution to the problem of how we will communicate the grid to the computer. We will use the fact that computers handle tables of numbers very well. Mathematicians use the word matrix for a rectangular table. The **grid matrix** will have as many rows and columns as the grid. We will put a 1 in the ith row and jth column of the table if there is a brace at the same place in the grid. If there is no brace at that spot in the grid we put a 0 in the corresponding place in the table. For example M is the grid matrix for the following grid:



$$M = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

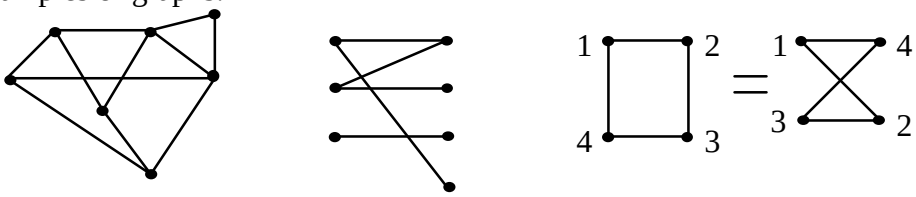
ACTIVITY 7

- What does a row of zeros in the grid matrix tell us about the grid?
- What does a column of ones in the grid matrix tell us about the grid?
- Restate the EROC (Empty Row or Column) Principle in matrix form.
- Let e = [1 1 1 ... 1] and e' be a similar column matrix (vector) with all 1's. What does eM and Me' (for appropriately sized e and e') tell about the grid? Here M is any grid matrix.

The idea of the grid matrix certainly solves, in a very easy way, the problem of communicating the pattern of the grid to the computer. More importantly we have opened the door to a whole new world. Mathematicians do all kinds of neat things with matrices (multiply, determinant, etc.). Since computers can quickly and accurately perform many matrix operations, we have brought into play the power of matrix algebra.

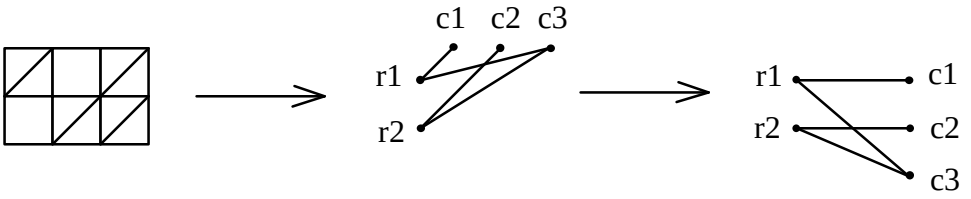
4.2 THE GRID GRAPH

Another powerful technique is to recast the grid rigidity problem into what is called graph theory. This branch of mathematics uses the word graph in a completely different way than when we refer to the graph of a function such as $f(x) = 5x^2 + x - 3$. In this new use of the word, a graph will look like a network of lines (called edges) which connect various points (called vertices). Following are three examples of graphs:



How we arrange the vertices on the page doesn't matter. Also, even though the edges sometimes look like they cross in the middle there is not a vertex at that point.

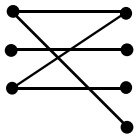
The following shows how we associate a graph with a grid:



grid grid graph grid graph redrawn in "two column" form

Note that the **grid graph** has vertices $\{r_1, r_2, \ldots, r_m, c_1, c_2, \ldots, c_n\}$; one for each of the m rows and one for each of the n columns of the grid. For example a 4 by 5 grid will have 9 vertices (m is 4 and n is 5). There is an edge between vertex r_i and vertex c_j precisely when there is a brace in the row i and column j position. We will draw our grid graph by putting the vertices in two vertical stacks; the row vertices on the left and the column vertices on the right. A brace then becomes an edge joining a vertex on the left with a vertex on the right. You should note that our grid graphs can't have any vertical edges, i.e. no edge connects two row or two column vertices.

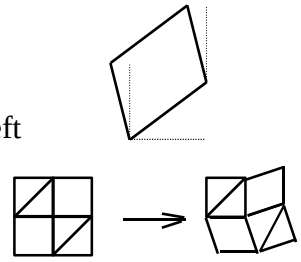
ACTIVITY 8



- Draw the grid corresponding to the following grid graph:
  Is this grid rigid or floppy?
- Write down a grid with no braces in the first row and draw its grid graph.
- Write down a grid with no braces in the first column and draw its grid graph.
- Restate the EROC (Empty Row or Column) Principle in graph form.

4.3 GRIDS REPRESENTED AS A SYSTEM OF LINEAR EQUATIONS

We briefly indicate the basic idea of another way to think about rigidity in a grid. If you are interested in a more complete treatment of this idea see the UMAP module by Brigitte Servatius, pages 3-5. To motivate the idea look at how a floppy grid can deform.



Notice that the orientation of any given square can be completely determined by two angles: the angle the sides make with the vertical and the angle the top-bottom makes with the horizontal. If we keep the top left edge vertical then in a rigid grid all of the angles must be zero. The situation is different for a floppy grid, but even here these angles are constrained. For example, by the Persistent Parallels lemma all the side angles in each row must be the same. Similarly all the top and bottom angles must be the same in each column. The presence of a brace causes the row angle and the column angle to be the same. If we let s_i be the angle the sides make with the vertical in row i, and b_j be the angle the bottom makes with the horizontal in column j, we can write a system of equations associated with any grid. The grid is rigid if and only if the system has no parameters (unique solution: all angles = 0).

s1 = 0	not unique
s1 = b1	solution, so
s2 = b2	floppy

Thus we have transferred the rigidity problem into an algebra problem, the solution of systems of linear equations. We can use what we know from algebra to help decide if a grid is rigid or how a floppy grid can be deformed. See Section 7.5 for a project extending this approach.

4.4 FOUR WAYS OF LOOKING AT THE RIGIDITY PROBLEM

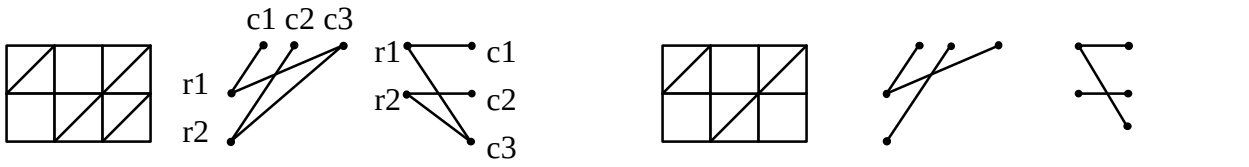
We now have four ways of thinking about rigidity in grids and each way has its own set of tools:

- as the actual grid (tool: geometry),
- as the grid matrix (tool: matrix and linear algebra),
- as the grid graph (tool: graph theory),
- as a system of linear equations (tool: algebra).

By recasting the problem in these different ways we have available the powerful tools that have been developed in each of these settings. To reiterate what we said at the beginning of this section, this is one of the most profound ideas of mathematics. In the following sections you will learn about the matrix and graph settings of the rigidity problem.

SECTION 5: GRIDS AND GRAPHS

You have seen in Section 4 that we can represent any grid with its corresponding grid graph. Here are two examples - the first is rigid, the second is floppy.



rigid grid grid graph graph redrawn floppy grid grid graph graph redrawn

The resulting graphs are **bipartite**: the vertices are naturally separated into two groups (the rows and the columns) such that edges only connect vertices that are in different groups.

5.1 HOW THE GRAPH SHOWS IF THE GRID IS RIGID

We run into a nice bit of luck here. The grid property of being rigid or floppy corresponds to a graph property that is well known and can be easy to check "by eye". Let's look at examples of grids and their graphs and see whether we can make the connection (pun intended). In the process we will review some of our grid conditions and practice the translation from grids to graphs.

ACTIVITY 9

i) Draw the graphs for the 2 by 3, 2 by 4, 3 by 3 , and 3 by 4 Gamma grids (they look like Γ). Recall these are all rigid.

ii) Do the same for the same size checkerboard grids. Recall these are all floppy.

iii) Draw graphs for several grid examples that have an empty row or column. Recall EROC says these will always be floppy.

iv) Draw graphs for the 2 by 3, 3 by 3, 3 by 4 and 4 by 4 staircase grids (start with a brace in the top left and repeat braces to the right and down). These are all rigid by the Builder's lemma. What happens with the 2 by 4 staircase?

Make a conjecture about the graph property tied to rigid/floppy grids. Test it on a few additional grids which we know to be rigid or floppy and see whether your conjecture stands the test.

Have you given yourself a fair chance at discovering the answer by doing activity 9?

Comparing the two graphs given at the beginning of the section, the major difference is that in the first graph (rigid grid) every vertex can be reached from every other vertex. A graph with this property is called **connected**. The second graph (floppy grid) is **disconnected** as you cannot reach c3 from r2 for example. We have thus discovered the main theorem in this section:

CONNECTIVITY THEOREM

A grid is rigid if and only if its grid graph is connected.

Notice that the graph being connected is both a necessary and sufficient condition. That is, both directions of the implication are true: if the grid is rigid, then the graph is connected, AND if the graph is connected, then the grid is rigid. To go from the current status of a conjecture or claim based on examples to a real theorem we need a proof.

The proof is based on the Persistent Parallels (PP) algorithm we developed in section 3. Briefly stated, PP started at the top left marking a vertical edge in the grid. Then all the other edges in that row had to stay vertical by the persistent parallels lemma. A brace in the first row allowed us to conclude that the braced square's top edge was horizontal, and so all the edges in that column were also horizontal. In this way we used braces to move from rows to columns to rows, and tried to get all grid edges horizontal or vertical.

Here is an activity to help you understand the proof. It has you review how to do the PP algorithm, and at the same time see what happens on the graph.

ACTIVITY 10
Use the two grids given at the start of this section and their grid graphs. Apply the PP algorithm developed in section 3 on each grid. When you use a brace to switch from horizontal to vertical marking (and vice versa) mark the corresponding graph edge. Note how we are constructing a path from vertex r1 to all the other vertices in the rigid example, and we can't do that in the floppy example.

Here is the formal proof.

i) If the grid is rigid, then the graph is connected

Since the grid is rigid, we know that PP will mark all horizontals and verticals, starting from the upper left corner. It follows that there is a sequence of braces (corresponding to edges in the graph) that connects row 1 to every other row or column. Likewise, there is a path connecting vertex r1 to every other row or column vertex. This ensures that there is a path from every vertex to every other vertex; for example, to go from r2 to c3, use the path from r1 to r2 (traveling backwards) and then the path from r1 to c3. This may not be the shortest path, but we know there is at least this path.

ii) If the graph is connected, then the grid is rigid.

Because the graph is connected there is a path of edges from vertex r1 to every other row and column vertex. These edges correspond to braces that allow the PP algorithm to switch from marking verticals to marking horizontals and vice versa. Thus PP will be able to mark all the grid edges and the grid will be rigid.

5.2 ALGORITHM FOR GRAPH CONNECTIVITY

To make this result really useful we need a way of telling if a graph is connected. For small examples we can tell just by looking, but with larger graphs it can be tricky. As is our pattern, we ask you to try to discover an algorithm on your own before we tell you someone else's answer.

ACTIVITY 11 (FIRST PART)

i) Form the graph for Monster Grid #1.
ii) Decide whether or not it is connected. Explain to another student how you decided.
iii) Write out an algorithm for checking connectivity based on your method. Try your algorithm on several of the earlier examples for grid graphs, e.g. the 6 by 8 checkerboard and the 8 by 10 gamma, to see whether your algorithm correctly identifies the rigid grids by declaring their grid graphs connected.

Here is an algorithm suggested by Brigitte Servatius [UMAP]. It reads a bit like a computer program, and could be translated to one (this is called pseudo-code). It also illustrates the feature of a good algorithm of using an iterative scheme and having a clear stopping condition.

Algorithm to check for connectivity

Step 0: Pick any vertex, circle it, and call it the active vertex.

Step 1: Circle all the vertices connected by an edge to the active vertex. Cross out the active vertex.

Step 2: If all the vertices are circled, then declare the graph connected and STOP.

Step 3: If not, and all the circled vertices have crosses, then declare the graph to be disconnected and STOP.

Step 4: Otherwise, declare one of the uncrossed circled vertices the active vertex. GOTO Step 1.

ACTIVITY 11 (CONTINUED)

i) Use the algorithm above to determine that the Monster Grid #1 graph is not connected.

ii) How many "connected components" does MG#1 have?

iii) Decide how many more braces are needed to make MG#1 rigid, and where they might go.

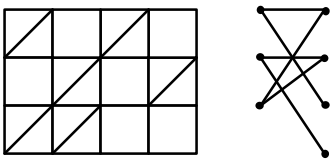
[If you wish only a quick overview of this paper, you can skip on to section 6. However, you will be missing some really neat applications of graph theory to our grid bracing situation.]

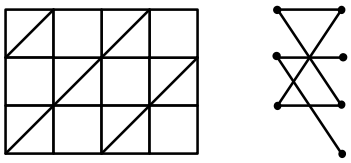
5.3 REDUNDANT BRACES AND CYCLES IN GRAPHS


—can fill in


redundant

We have some feeling for when a brace is needed and when it may be redundant. For example, the Builder's algorithm allows us to fill in the fourth brace in a 2 by 2 subgrid with three braces present. So if the fourth brace is already there, it is redundant. In fact, any one of the four is redundant. Any one of the four can be removed without affecting the rigidity of the grid. Redundant braces show up nicely on the graph as "cycles". We illustrate these ideas with examples before we give the precise definitions. Here are two examples of 3 by 4 grids with m+n-1 = 6 braces (the minimum needed for rigidity).


rigid


floppy

Note they differ only by the placement of one brace in the third row.

Recall by the Connectivity Theorem the grid is rigid if and only if the graph is connected. Do you see that the first graph is connected - it is actually a single path of length 6. The floppy grid is the 3 by 4 checkerboard - do you see that its graph is not connected?

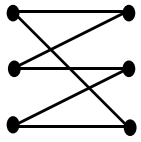
The new idea we wish to point out has to do with how redundant braces show up in the graph. The second graph has a "**cycle**", a simple path that begins and ends at the same vertex. (A simple

path can not use any edge more than once.) Here the cycle is r1-c1-r3-c3-r1. It shows up on the grid as a (nonadjacent) 2 by 2 with all 4 braces, and on the graph as a figure 8 shape. Such a cycle introduces redundancy into the graph as it allows more than one way to go from any one vertex in the cycle to another. Because the checkerboard example has a redundant brace it has effectively only five braces, one short of the m+n-1 = 6 needed for rigidity.

ACTIVITY 12

1) Draw the 4 by 4 checkerboard grid and its graph. How many cases of "full" 2 by 2 subgrids do you see on the grid? How many figure 8's do you see on the graph? If you remove redundant braces how many more are needed to make the grid rigid? On the graph if you remove an edge from each cycle, how many new edges are needed to make the graph connected? Now draw the 5 by 5 checkerboard grid and graph and answer the same questions for them.

2) We show a bipartite graph with edges making a "6-cycle". Draw the corresponding grid. Verify that the grid is rigid (i.e. that the graph is connected), and that any <u>one</u> brace/edge can be removed without losing those properties (grid rigid/graph connected). Note it is not as easy to spot this redundancy on the grid or the graph as it was for a 4-cycle. But it is easier to see on the graph than on the grid.



3) Cycles are harder to see as they get longer. What is the longest cycle in a 5 by 8 checkerboard? ... in an m by n checkerboard?

We can exploit this connection between redundant braces and cycles to give a formal definition of redundant braces.

DEFINITION

A grid brace is a **redundant brace** if the corresponding edge in the grid graph is part of a cycle.

Can you supply an argument for the following claim? If a redundant brace is removed from a rigid grid, the grid will still be rigid. The argument we have in mind is based on showing the graph will remain connected. Note that all the braces corresponding to edges forming a cycle are redundant by our definition. However, if one is removed, then the rest may well not be redundant any more.

5.4 MINIMAL BRACING AND SPANNING TREES

The last idea we will explore in this section is that of a "minimal bracing" and its graph translation as a "spanning tree". These ideas come out of our M+N-1 claim (as yet unproved) that says in any m by n grid you need m+n-1 braces for rigidity. A related question is whether every rigid grid can be stripped of redundant braces to end up with m+n-1 braces. We will take advantage of known results from graph theory verify our M+N-1 claim and to answer the question with yes.

This time we will proceed right to the relevant definitions and quote the needed theorem from graph theory. The work we have done so far allows us to apply that theorem in the grid context.

DEFINITION

1) A **minimal bracing** of a rigid grid is a subset of the braces which keeps the grid rigid but will become floppy with the removal of any of its braces.

2) A **spanning tree** of a connected graph is a subgraph containing all the vertices which is connected and which has no cycles.

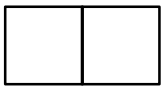
GRAPH/TREE THEOREM
1) Every connected graph has (potentially many) spanning trees. Any spanning tree has a minimal property that removing any edge will disconnect the graph.
2) Every tree with v vertices has v-1 edges

COROLLARY (A "corollary" is a result of special interest that follows directly from a theorem.)
1) Every rigid m by n grid has (potentially many) minimal bracings. These will be rigid and removing any edge will make them floppy.
2) Every minimal bracing of an m by n grid has m+n-1 braces.

Note that by translating our grid rigidity problem into the graph context we have gained a BIG advantage. Not only are things often easier to see on the graph than the grid, but also very much is known about graphs.

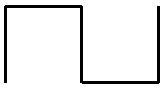
ACTIVITY 13

i) We show a graph to the right. Verify it is bipartite and re-draw it in "two column" form. Draw its corresponding grid.



bipartite graph

ii) We show a spanning tree. Find 3 different spanning trees of the original graph. Translate each to a minimal bracing of the grid.



one spanning tree

iii) How many spanning trees are there?

There are standard algorithms for finding spanning trees. Some add vertices and edges while avoiding cycles, and others strip away edges while maintaining connectivity. Here is one that is deceptively simple (it is not as carefully written as the connectivity one).

Tree-Growing Algorithm

Step 0: V = the set of vertices in tree: initially pick any vertex, v_o, in the graph and set V = $\{v_o\}$. E = the set of edges in tree: initially E = $\varnothing$.

Step 1: Select any vertex v in V . As long as there are edges {v, w} in the graph such that v is in V and w is not in V, add {v, w} to the edge set E and w to the vertex set V.

Step 2: Repeat step 1 under some systemic procedure that marches through all the vertices in V. If V contains all the vertices in the graph, then a spanning tree has been found. Otherwise there is no spanning tree for the graph because it is not connected.

We can use this algorithm to produce a spanning tree for a given graph. In most cases, the spanning tree will not contain all the edges of the graph. Stop reading at the end of this sentence and think about what it means for an edge to be outside the spanning tree. [Here is our answer - have you thought of your own answer?] It means those edges are redundant.

This algorithm and the graph connectivity one can be programmed for a computer. See section 7.3 if you are interested in doing programming. Both Mathematica and Matlab come with built-in functions related to graph theory. In Appendix II.we describe how to use these.

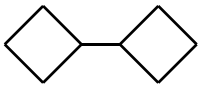
EXERCISES FOR SECTION 5

5.1 Describe how a case of EROC and of OBIRAC show up on the graph. Will such a graph be disconnected? (Find the "trivial" case that is exceptional.)

5.2 i) We have conjectured that none* of the checkerboard grids is rigid. Explain that with the connectivity theorem. That is, show that no* checkerboard grid graph is connected. (*The 1 by 1 is an exception - is this the only exception?)

ii) Show with an example that the following conjecture is false, and refine it into a true one. "Every checkerboard grid can be made rigid by adding any one additional brace."

5.3 i) We show a graph with a "bridge" - an edge whose removal will disconnect the graph. Verify that this is a bipartite graph, redraw it in "two column form", and draw the corresponding grid Explain why every edge but the bridge is redundant.



ii) Explain why every edge in a spanning tree is a bridge.

5.4 Define "redundant brace" in grid terms. (Recall we defined it in terms of the graph.)

5.5 Define "connected component" of a graph. Find a way of relating the number of additional braces needed to make a floppy grid rigid to the number of connected components of its graph. That is, make a conjecture based on looking at examples - prove it if possible.

5.6i) Discover the idea of the number of "independent" redundant braces. For example, the 2 by 2 grid with all 4 braces has 1 independent redundant brace. The 2 by 4 grid with the first column empty and the other columns fully braced has 2 independent redundant braces. How many independent redundant braces does the 3 by 4 checkerboard have? ...the m by n checkerboard:

ii) Develop an algorithm that will take any grid, rigid or floppy, and tell how many independent redundant braces it has.

5.7 In playing the bracing game in the building version we sometimes need to know if a particular brace we might add would be redundant or not. The grid at the time would be floppy. In the gutting version we are seeing a rigid grid and need to know if the brace we are thinking of removing is redundant or not.

i) Explain how to do this with the grid graph.

ii) Develop a way of answering based just on the grid

5.8 Using the ideas of this section (redundant braces, rigid = connected) explain how to play the bracing game optimally. For example, if both players play optimally, will the first or second player win the 3 by 4 building version? ... the 3 by 4 gutting version?

5.9 Write an algorithm to take any connected graph and prune it to a spanning tree.

5.10 What is the length of the longest possible path from the row 1 vertex to any column vertex in a 4 by 7 grid graph? ... in an m by n grid graph where $m \leq n$?

5.11 Make a conjecture that ties the length of the longest path from the row 1 vertex to any column vertex with the number of "turns" in the PP algorithm, and/or with the number of iterations needed in the BB algorithm. Check your conjecture on a variety of rigid and floppy grids of various sizes.

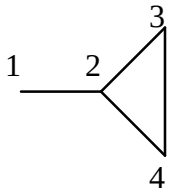
SECTION 6 MATRICES

In this section we will see how matrices can be used to determine if a grid is rigid. We look first at the adjacency matrix of the grid graph. Then we take full advantage of our special bipartite graph situation to cut the computations required by more than 75%. Even with that efficiency the matrix approach is cumbersome to do by hand. However it does allow us to apply another set of mathematical tools, and is the best way for a computer to determine grid rigidity.

We assume you know how to add and multiply matrices, that the row number comes first in references (e.g. the 1,2 entry is in row 1 and column 2) , that the order of matrix multiplication matters, etc.

6.1 THE ADJACENCY MATRIX, PATHS AND GRAPH CONNECTIVITY

The matrix that will help us is the **adjacency matrix** of a graph. Here is an example showing a (non-bipartite) graph and its adjacency matrix.



$$A = \begin{matrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{matrix}$$

The adjacency matrix A has as many rows and columns as the graph has vertices. It has a 1 in row i and column j if vertex i and j are adjacent (and a 0 if not). Because our graph is not "**directed**" (meaning the edges have no direction - they go both ways) the matrix A is **symmetric** (the i,j element and the j,i element are the same).

One of the surprising and nice uses of the adjacency matrix is that its powers give the number of paths of a given length between the vertices. Here are the powers of A, the adjacency matrix for the graph above.

$$A = \begin{matrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{matrix} \qquad A^2 = \begin{matrix} 1 & 0 & 1 & 1 \\ 0 & 3 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{matrix} \qquad A^3 = \begin{matrix} 0 & 3 & 1 & 1 \\ 3 & 2 & 4 & 3 \\ 1 & 4 & 2 & 3 \\ 1 & 4 & 3 & 2 \end{matrix}$$

Each of these powers should be symmetric - meaning every i,j and j,i entry are the same. You can also spot symmetry because the top right triangle of entries above the main diagonal should reflect across the main diagonal to give the bottom left triangle. A third equivalent meaning is A should equal its **transpose**, written $A = A^t$ (sometimes written A'). The transpose matrix is formed from any matrix by turning its rows into columns (or vice versa). Note A^3 as we have it is not symmetric. Can you see why? Right, the 2,4 and 4,2 entries are different. Check which one is right by counting the paths of length 3 from vertex 2 to 4 in the graph. For example, 2-1-2-4 is one such path. Double check by multiplying row 2 of A times column 4 of A^2, and similarly finding the 4,2 entry by multiplying.

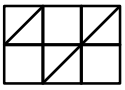
The use we have for this path counting is to find out if a grid graph is connected. Recall the important result from last section that the grid is rigid if and only if its graph is connected. Normally in graph theory we add the powers of the adjacency matrix for this test of connectivity. If all entries are nonzero in a large enough sum of powers, that means there is at least one path from each vertex to every other. The highest power needed is one less than the number of vertices because that is the longest simple path in a spanning tree of the graph. We show in the next subsection how to get by with a much smaller computation for our special (bipartite) grid graphs.

First Matrix Test for Rigidity
Let A be the adjacency matrix for the grid graph of an m by n grid. Let P be the sum of the powers of A (stop with the m+n-1 power).
The grid is rigid if and only if P has no 0 entries.

Three notes:
- We will improve on the computational efficiency of this in the next subsection. However some math software tools automatically find A and its powers (see Appendix II), so this has some usefulness..
- If you wish to be *very* careful, the result as stated is wrong for the 1 by 1 grid (check that - how about the 1 by 2 grid?). It can be fixed by adding in the 0th power of A, but we will forgo that bit of precision.
- The way we number the vertices in a graph *does* affect the adjacency matrix, but not the connectivity result. For convenience, we will number the row vertices first, followed by the column vertices both in their natural order.

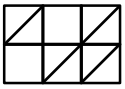
ACTIVITY 14

i) For the grid shown form the grid matrix, M, the 2 by 3 matrix with 1's where the braces are (and 0's where they aren't). Calculate M', MM', M'M, and decide which of these are symmetric.



ii) For the same grid, draw the grid graph and find its 5 by 5 adjacency matrix. Note the grid is not rigid by OBIRAC. Calculate the sum of the powers of A ($A + A^2 + A^3 + A^4$) to see both how 0 entries persist, and what a pain it is to do all that matrix arithmetic. (Get help from a machine if possible.)

iii) Repeat parts i) and ii) for the rigid 2 by 3 grid shown. Focus on how the grid matrix, M, and its transpose show up in the adjacency matrix.



6.2 SHORTCUT MATRIX TEST FOR GRID RIGIDITY

Here is the best way we know to use matrices to test rigidity. The rest of this subsection attempts to motivate and explain why it works. You are forgiven if you forgo that for now - it is really neat though, and involves block matrices, and some clever ways to exploit the bipartite nature of grid graphs.

Best Matrix Test for Grid Rigidity (BMT)
Let M be the grid matrix for an m by n grid ($m \le n$). Let Q be the matrix product MM'MM'...M having 2m-1 factors. Note 2m-1 is odd so the product will begin and end with M. Note also Q will be the same m by n size as M.
The grid is rigid if and only if Q has no 0 entries.

Some quick notes before you dash off to Activity 13
- For a 3 by 3 grid, Q = MM'MM'M. We calculate a product of 5 factors, each a 3 by 3 matrix. Compare this with the calculation of $A+A^2+...+A^5$ where A is the 6 by 6 adjacency matrix. We realize a 75% savings in storage and computation. In a 3 by 10 matrix the savings are very substantially larger. Check that out.
- The BMT says if Q has any 0 entries, the grid is not rigid. The position of the 0's has significance for how to add braces to make the grid rigid. You can explore that in exercise 6.4.

ACTIVITY 15

1i) Here is the adjacency matrix for a grid graph. Recognize its block form and reconstruct the grid (how many rows? ___, how many columns? ___, what is M?, Draw the grid).

$$\mathbf{A} = \begin{matrix} 0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&1&1&1&0 \\ 0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&1&0&1&0&0&0&0&0 \\ 0&1&1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&0 \end{matrix}$$

ii) Decide if the grid is rigid by PP or BB. The BMT says to calculate a product of 7 factors of M and M'. See if you can get all nonzero entries with the 3-product or the 5-product. (Calculate MM'M and MM'MM'M.) Does this somehow correspond to how many iterations of Builders or how many "turns" of PP are needed?

2 Construct the grid matrix for the 3 by 4 Γ-grid and the 3 by 4 checkerboard grid. Get help from some software and calculate the Q matrix for each. Check that the results are what you know to expect (Γ is rigid, checkerboard isn't).

For the remainder of this section we exploit the special (bipartite) nature of a grid graph to explain and illustrate the Best Matrix Test for Grid Rigidity.

LEMMA 1
The grid matrix M shows up as a block in the adjacency matrix A, and powers of A involve products of M and M'.

Revisit Activity 14 and note how the grid matrix, M, and its transpose, M', show up in the adjacency matrix A. Further note how MM' and M'M show up in A^2. What we have here is a nice application of what are called **block matrices**. We can quickly calculate the powers of A by taking advantage of this block structure. Here is how things look for a general m by n grid(the subscripts give the size of each block).

$$A_{(m+n),(m+n)} = \begin{matrix} O_{m,m} & M_{m,n} \\ M^{t}_{n,m} & O_{n,n} \end{matrix} \qquad A^2 = \begin{matrix} MM'_{m,m} & O_{m,n} \\ O_{n,m} & M'M_{n,n} \end{matrix} \qquad A^3 = \begin{matrix} O & MM'M \\ M'MM' & O \end{matrix}$$

The blocks of zeros in A correspond to the fact that there are no paths of length 1 from a row to a row (or column to column). The blocks of zeros in A^2 reflect no paths of length 2 from a row to a column and vice versa. Similarly there will be such blocks of zeros in all the powers of A. And in the odd products the upper right block matrix will be a product of the form MM'MM'...M. Thus the entries of such products count paths from rows to columns in the grid graph. We state this result formally as a corollary. Here the theorem this follows from is that entries of powers of the adjacency matrix count paths.

COROLLARY
Let R be any odd product of M and M' matrices: R = MM'MM'...M with 2k-1 factors. The i,j entry of R gives the number of paths of length 2k-1 from the row i vertex to the column j vertex in the grid graph.

LEMMA 2
Each entry either increases or stays the same as we increase the length of these odd product MM' matrices.
This follows because the entries count paths from a row to a column. Any such path (which must be of odd length) can be extended to a path 2 steps longer by just repeating the last two steps. For example, back in our first graph the path 2-1-2-4 of length three can be expanded to 2-1-2-4-2-4, a path of length 5. Thus there are at least as many paths of length 2k+1 as there were of length 2k-1. In particular, once any entry is positive, it stays positive.

THEOREM
For a grid graph to be connected it is necessary and sufficient that the 2m-1 product MM' matrix has no 0's (i.e. we don't need to add powers like we do with A)

Proof If the grid graph is connected then there is a path from every row vertex to every column vertex. Since every path must alternate from a row vertex to a column vertex and there are only m row vertices, the longest such a path can be is 2m-1. Thus by lemma 2 all the entries of the 2m-1 product MM' matrix will be positive.

For the converse we start by knowing there is a path from every row vertex to every column vertex. To show the graph is connected we need to show there is a path from every vertex to every other vertex. For example to get a path from row 1 to row 2, stitch together a path from row 1 to column 1 with a path from row 2 to column 1. Similarly we can get a path between any two rows or two columns.

EXERCISES FOR SECTION 6

6.1 Verify the matrix form of EROC. That is, show if a grid matrix has a row or column of 0's, then the Q matrix will have some 0 entries. Do the same for OBIRAC. (Be specific about which entries will be 0 and why.)

6.2 Get help from a machine and calculate the Q matrix for a variety of grids. Include at least two sizes of checkerboards and of gammas. Verify that BMT works in these cases.

6.3 Tackle the Monster Grid #1 using BMT. Does Q have 0's? Does their position tell you where to add braces to make the grid rigid? (See Activity 11.)

6.4 When a given grid is floppy we might want to make it rigid by adding as few braces as possible. Here we ask you to check out the conjecture that you will need to add braces wherever there are 0's in the Q matrix defined above. If it is a false conjecture, not only give a counterexample, but try to refine it into a good result. Start by looking at the 2 by 3 checkerboard grid. Is it rigid? (no) How many 0's appear in the Q matrix? How many braces can we get away with to make it rigid?

6.5 Try to tie the number of 0's in the Q matrix with the minimal number of braces needed to make a floppy grid rigid. This is directly related to the number of connected components of the grid graph (see exercise 5.4).

6.6 Check out the conjecture that in a rigid grid matrix the number of 0's in the odd product MM' matrices must be strictly decreasing. That is, if at any stage the number of 0's doesn't decrease, then the grid is floppy.

6.7 The Q matrix will usually have nonzero entries where the corresponding square was not braced. Check out this conjecture: if we add a brace to such a square, it will be redundant. If this is true, it would help us play the bracing game by identifying where we can "stall" by adding redundant braces.

6.8 Formulate and check out a conjecture that we can test for rigidity by just looking at powers of the MM' matrix. Can MM' have 0's persist in its powers even though the grid is rigid? If MM' has no 0's does that imply the grid is rigid? What is the minimum power of MM' we would need to check?

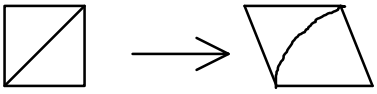
6.9 Try to relate our BMT matrix property with some known matrix properties. For example, will the determinant always be nonzero for a rigid grid matrix (this needs a square grid)? Does the trace get involved? Is there anything special about the eigenvalues for rigid grid matrices? Instead of looking at M, should we look at MM' for these questions?

SECTION 7 SUGGESTIONS FOR FURTHER EXPLORATIONS AND REFERENCES

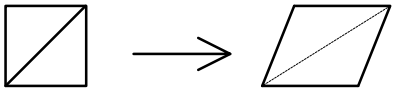
This is a bit of a miscellaneous collection of loose ends and ideas that came up in our discussions that we have not taken the time to organize or explore ourselves. (Except 7.6 which involves counting and probability is better developed.) We also include a brief list of references with comments on how you might use them to follow up these ideas on your own.

7.1 BRACING WITH CABLES - NON RIGID DIAGONALS

Often in physical settings the diagonals used to do the bracing are not stiff like metal beams, but are ropes or cables. They will not stretch, so they will prevent one direction of deformation, but will allow flexing in another.



YES
cable can
bend



NO
cable can't
stretch

We claim you will need both diagonal cables to make the 1 by 1, 1 by 2, ... and in general the 1 by n grid rigid. Further, given any arrangement of stiff diagonal braces that makes the grid rigid, we can keep it rigid by putting two crossing cables in each braced square. The interesting question is whether we can make a grid rigid with cables and use fewer than twice the number of stiff diagonal braces.

PROJECT 1 (these are more extensive and less directed explorations than the Activities)

1.1 Give an argument that a 2 by 2 with cables as shown is rigid. Verify this uses two less than twice what it would take with our earlier stiff braces. Explore lots of other small grids to see how much better you can do than twice m+n-1. Aim for a general result like "An m by n grid can be made rigid with _____ cables"

1.2 See how our two grid rigidity tests (Persistent Parallels and Better Builders) might be modified to work with cable braced grids.

1.3 See how the graph version of the problem might be modified to work here with cable braced grids. The graph theoretic ideas of directed graphs (= digraphs), and strongly connected digraphs will almost surely play a role here.

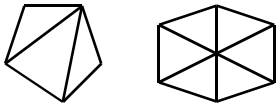
1.4 See how the matrix version of the problem might be modified here. Again it might speed your investigation to look up how adjacency matrices work for digraphs.

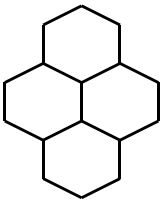
7.2 BRACING OTHER SHAPES

We started with the observation that although triangles are inherently rigid, squares are not. Squares form up into grids in a natural way so our interest has been in making (square) grids rigid. We suggest two different problems that deal with other shapes. One looks at stiffening a single n-gon for $n > 4$, and the second looks at other grids (tesselations).

PROJECT 2

2.1 Verify that any two diagonal braces (we will assume stiff braces) will make a pentagon rigid, and no single brace will.
Investigate the same kind of question for hexagons. That is, how many diagonal braces does it take, and does every arrangement of that many diagonals work? For example, if you have all three "long" diagonals in the hexagon as shown, is it rigid?
Continue this kind of question for 7-gons, 8-gons, etc.





2.2 There are only three kinds of "regular tesselations" of the plane. One uses triangles and is inherently rigid. We have already considered the one that uses squares. The third uses hexagons. Develop a theory of rigidity of hexagonal grids. Look up what semiregular tesselations are, and investigate what it takes to make them rigid.

7.3 MORE ON THE BRACING GAME

We seek a complete analysis of the small size games. For example, the first player will always win the building 2 by 2 game and the second player will always win the gutting 2 by 2 game. Who will always win the 2 by 3 games in each version? Is that automatic or affected by the player's strategy?

PROJECT 3

3.1 Discover and describe a winning strategy for each game. How can you decide when to add redundant braces versus when you should move efficiently toward a rigid grid? If both players play well, is the winner automatically determined by the size of the grid? Is the gutting version substantially different in this respect? If we start the game with some random set of braces already in place, find a way to determine if you would like to play first or second.

3.2 If you like computer programming, write a program to play the game - perhaps a smart and a dumb (random) version. Perhaps just write a program to have the computer decide if the grid is rigid by one of the algorithms.

7.4 EXPLORING SYMMETRIES AND GRID EQUIVALENCE

If we rotate or reflect a given grid the result is in some sense the "same". For example, if the original grid was rigid then so is the rotated or reflected grid. Rotation and reflection are examples of what we call **symmetries**, transformations that preserve important properties of geometric figures (in this case distance is what is being preserved).

PROJECT 4

Explore what other kinds of changes we can perform on a grid (besides rotation and reflection) and still maintain rigidity and floppiness. For example, can we reorder the rows and still have the "same" grid? (Convince yourself the answer is yes.)

Investigate how many "different" grids there are of a given size and number of braces. For example using just reflection there is just one 2 by 2 grid with each number of braces. How many really different 2 by 3 grids are there? Are all 3 by 3 grids with 5 braces the same under these allowable transformations?

7.5 HOW FLOPPY IS FLOPPY?

The unbraced 1 by 1 square has "one degree of freedom" in that if we nail down the bottom edge, each top (floppy) corner can only move along a semicircle (a one dimensional locus). Further they cannot move independently of each other. If we nail down the bottom left edge of an empty 1 by 2 grid, the locus of the top right corner is two dimensional. It has area. (It is a very interesting shape.) Also that top right corner can move somewhat independently of the top left corner. So we say the unbraced 1 by 2 has two degrees of freedom.

PROJECT 5
Explore the ideas of degrees of freedom in larger grids. Does the 2 by 2 with two braces have one or two degrees of freedom? Should the idea of degrees of freedom be tied to independence of movement of corners or to dimension of the loci? Maybe tie this in to the linear algebra ideas of rank and dimension of the solution space for the system of equations approach mentioned very briefly in section 4.

Explore the locus of all the corners in a floppy grid. Look up what a pantograph is, and tie it to these investigations.

7.6 HOW LIKELY ARE GRIDS TO BE RIGID?

We know that in an m by n grid we must have at least m+n-1 braces before the grid can be rigid. In this section we will investigate how likely it is for a bracing to be rigid when we use exactly m+n-1 braces. For example, the 2 by 3 grid needs 4 braces: if we pick their positions randomly, how likely is it that the resulting grid is rigid?

We will need Pascal's triangle and the fact that its numbers are combination numbers that count the number of ways to choose k different objects out of n objects (where the order of choosing doesn't matter). Recall the formula $C_{n,k} = \frac{n!}{k!(n-k)!}$ gives the numbers in row n of Pascal's triangle. Check that $C_{6,0}$ $C_{6,1}$ $C_{6,2}$ ... $C_{6,6}$ does give the sixth row. We can use these numbers, $C_{mn,k}$, to count the number of ways to put k braces into an m by n grid. That is because we are choosing k squares out of the mn squares to put the braces in, and the order we choose the squares doesn't matter.

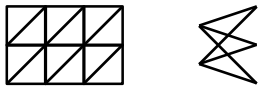
```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
1   6  15  20  15   6   1
```

In particular, there are $C_{6,4} = 15$ ways to put 4 braces into a 2 by 3 grid. How many are rigid? Here it is easier to count the ones that are floppy. There are 3 ways to put 4 braces into the 2 by 3 grid that leave a column empty. All the other 12 will be rigid. So if we put them in randomly, the probability is 12/15 = .8 that the grid will be rigid.

For a 3 by 3 grid the counting gets much harder. Here m+n-1 = 5, and there are $C_{9,5}$ = 126 ways to put in 5 braces. What is hard is counting how many are rigid (or how many are floppy). By a careful look at cases (row empty, column empty, OBIRAC) we came up with 45 floppy, so 81 are rigid. Thus the probability for the 3 by 3 grid with 5 braces to be rigid is 81/126 = .643

Because counting the rigid bracings gets so hard so fast we will tell you a clever and magic shortcut developed by graph theorists who wanted to count the number of spanning trees in a graph. The method uses the determinant of a strange submatrix derived from the adjacency matrix of the graph. Here is an example finding the 12 ways for a 2 by 3 grid.

We start with the completely full 2 by 3 grid and the corresponding graph. A is the associated adjacency matrix. We define a new matrix, D, whose only nonzero entries are on the diagonal. There we put the number of edges that come out of each vertex (graph theorists call this number the **degree** of the vertex).



$$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}, \qquad D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

We now define a new matrix: C = D - A. In the above example:

$$C = \begin{bmatrix} 3 & 0 & -1 & -1 & -1 \\ 0 & 3 & -1 & -1 & -1 \\ -1 & -1 & 2 & 0 & 0 \\ -1 & -1 & 0 & 2 & 0 \\ -1 & -1 & 0 & 0 & 2 \end{bmatrix}$$

THEOREM:
Form the C = D - A matrix for the m by n completely braced grid. The number of m+n-1 <u>rigid</u> bracings is given by the determinant of the submatrix of C obtained by deleting the first row and the first column.

In the above example: determinant(submatrix of C) $= \begin{vmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & 0 & 0 \\ -1 & 0 & 2 & 0 \\ -1 & 0 & 0 & 2 \end{vmatrix} = 12$

Thus there are 12 rigid 4-bracings for a 2 by 3 grid.

See Ken Bogart's book, pages 177-80, for more on this. The theorem is deep and the proof is beyond the level of graph theory we have developed. (For those of you who know about cofactors, any cofactor of C can be used. It is another amazing fact that all of the cofactors of C have the same value.)

Computing determinants by hand is tedious and is very error prone. Fortunately many calculators and computer math software such as Mathmatica, Mathcad, Derive, and Maple compute determinants very easily. In the following project you will want to use machine help.

PROJECT 6

Investigate how the size and shape of the grid affects the probability that a randomly chosen minimal (m+n-1) set of braces will make it rigid. Fill in appropriate parts of the table shown and try to see patterns. What happens as we move right in a row, what happens as we move down the diagonal, will the probabilities go to 0? ... reach some positive limit?, etc. Patterns might be easier to spot if number of rigid rather than probability of rigid are used.

n:	1	2	3	4	5	6
m: 1	1	1	__	__	__	__
2		1	.8	___	___	___
3			.643*	___	___	___
4				___	___	___
5					___	___
6						___

n:	1	2	3	4
m: 1	1	1	__	__
2		4	12	___
3			81*	___
4				___

* in particular, use the determinant theorem to verify our count of 81 rigid here (3 by 3 case)

7.7 REFERENCES

1. Brigitte Servatius, Graphs, Digraphs, and the Rigidity of Grids, published by COMAP, Inc. as UMAP Module 744,. Lexington MA, 1995

   This is a very nice treatment of the rigidity problem we have been considering with a slightly different emphasis. It has an excellent set of references.

2. Brigitte Servatius, Rigidity & Braced Grids, 1995, published by COMAP as part of its Geometry and its Applications series. ISSN 1071-6874

   This is essentially the same as the UMAP module above. COMAP's address is

   COMAP, Inc. (617) 862-7878
   Suite 210 or 800-77COMAP
   57 Bedford St.
   Lexington, MA 02173

3. Kenneth P. Bogart, Introductory Combinatorics, published by Pitman, Boston, 1983

   This is a good reference for the graph, digraph and tree material we have mentioned. There are many other books you could use for this purpose - look for "combinatorics", "graph theory" or "discrete math" in the title.

APPENDIX I ANSWERS/COMMENTS TO SOME ACTIVITIES AND EXERCISES

SECTION 1

Activity 1

i) The 2 by 2 with 3 braces is rigid. For an argument see page 6. ii) No
iii) No (but see Section 7.1).

Exercises 1

1.1 Each of the 1 by 2, 1 by 3, ... 1 by n grids needs all squares braced to be rigid.
1.2 See page 1 (top left corner) for one way to deform the given 2 by 2.
1.3 We can make any n-gon rigid with n-3 braces all joined at one vertex. They are rigid because they are made up of triangles. For hexagons and higher it is possible to have n-3 braces and still leave it floppy. (You find it.)

SECTION 2

Activity 2

i) The 2 by 2 with braces in row 1, column 1 and in row 2, column 2 works. It can be done with several 2 by 3's, 3 by 3's as well - see page 1 for a 3 by 3 example.
ii) •F •T •F iii) •T •F •F •T

Activity 3

iii) Second player can always win the 2 by 3 building game: first can win the 3 by 3. What about the gutting game?

Activity 4

For the unclear phrase "to go from" it needs to be understood that you can't go across on a diagonal, but need an edge in common. Further simple examples suggest it is not enough to go from right side to left side. A refined conjecture might be to work with the condition that you can get from side to side and top to bottom. Decide if that is necessary, sufficient, both or neither.

Exercises 2

2.1 Builder's gets nowhere with the checkerboard grids. The proper conclusion is no conclusion (we will show later they are all floppy). Staircase grids are rigid if n is not too much bigger than n (make that precise). All the gamma rids are rigid.
2.3 ii) The 3 by 4 checkerboard grid is floppy yet has no cases of OBIRAC.
iii) If a grid is not floppy (i.e. rigid), then it has no cases of OBIRAC.
2.4 ii) If an m by n grid has fewer braces than m+n-1, then it is floppy.
iii) One example is the 3 by 3 checkerboard: it has 5 braces yet is floppy.
2.5 Combination numbers, Pascal's triangle and powers of 2 come into play here. For example, there are 2^6 = 64 2 by 3 grids, and the sixth row of Pascal's triangle, 1 6 15 20 20 15 6 1, gives the number with 0 1 2 etc. braces. Looking at the 15 grids with 4 braces (that's the minimum number for rigidity), we find only 3 which are not rigid (they have an empty column). So the probability is 12/15.
2.6 Look ahead to Activity 6 for one example.

SECTION 3

Activity 5

It should work starting with any single edge nailed down or any single brace nailed down.

Activity 6

You should see that Builder's fails to decide rigidity because you can't fill in all the squares, and Persistent Parallels decides it is rigid because all sides get marked.

Exercises 3

3.1 b) (last question) No.

3.2 The result should be rigid no matter which brace gets moved!

3.3 The Persistent Parallel algorithm allows you to fill up the whole row of vertical edges when you get a must-be-vertical in that row. Similarly for the whole column of horizontals. Having three of the four squares braced in a non-adjacent 2 by 2 allows us to get the non-braced side edges vertical and the top and bottom edges horizontal.

SECTION 4

Activity 7

- There are no braces in the corresponding row of the grid; thus, the grid is floppy.
- That column of the grid is completely braced.
- Matrix EROC Principle: If a row or column of the grid matrix contains all zeros, then the grid is floppy.
- Try calculating eM, Me' and eMe' for several small grids. For example, if

$$M = \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix}$$ then eM = [1 1]M = [1 1 1}, Me' = $$M \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} = \begin{matrix} 2 \\ 1 \end{matrix}$$, and eMe' = 3

Activity 8

- The grid is floppy by OBIRAC.
- The top left vertex of the graph will not have any edges connected to it. Graph theorists say that this vertex is **isolated**.
- The top right vertex of the graph will not have any edges connected to it.
- Graph EROC Principle: If any vertex of the grid graph is isolated, then the grid is floppy.

SECTION 5

Activity 9

iv) The 2 by 4 staircase is floppy be EROC. Find a necessary and sufficient condition on m and n to assure the staircase grid is rigid.

Remember the point here is to find a property of the graph that reflects the grid being rigid.

Activity 11

first part Monster Grid #1 is not connected. Make sure your algorithm is clearly stated and unambiguous, is correct, and that there is a clear indication of how it stops.

Activity 11 continued

Monster Grid #1 has 3 connected components. The 8-cycle r4-c1-r5-c15-r8-c10-r6-c6-r4 is one of them. Two more braces are needed: for example, braces in 1,1 and 1,2 will connect the resulting graph. (There are many other ways to do that.)

Activity 12

i) The 4 by 4 has 2 full 2 by 2's and figure 8's. One brace is needed; one edge is needed. For the 5 by 5 the numbers are 4 and 1.

iii) The longest cycle in a 6 by 8 checkerboard is a 6-cycle. Same for a 5 by 8. In an m by n checkerboard the longest cycle is m (if m is even) or m+1 (if m is odd).

Activity 13

iii) Here is a false claim. Show it is false and come up with the correct answer. CLAIM: there are 21 spanning trees: choose any 2 of the 7 edges to erase.

Exercises 5

5.1 For OBIRAC the graph will show a row vertex and column vertex only connected to each other. If there are any other rows or columns the graph will be disconnected: The 1 by 1 grid with a brace is the exception - it is connected despite having a case of OBIRAC.

5.2 i) For all checkerboard grids the squares with an odd row+column total are never braced. So on the grid graph you can never get from an even row to an odd column or odd row to an even column. The 1 by 1 is the only exception because it has no even row or column.

ii) Made true by requiring m > 1 (it is false for the 1 by 4 for example).

5.3 i) Every edge that is part of a cycle is redundant.

ii) By the minimal property of a spanning tree the removal of any edge will disconnect the graph (when you remove an edge you don't remove any vertices).

5.4 This seems difficult. One approach might be to say a brace is redundant if it is not used in an application of the PP algorithm. Check out if this works.

5.5 The idea is that a connected component is connected and cannot be made any larger (more vertices) and stay connected. You can connect a graph with k connected components with k-1 additional edges.

5.6 The 3 by 4 checkerboard has 2; the m by n has ... (tough problem - look at a bunch of examples: tie in with the magic number and the number of connected components).

5.7 This is easy to do on the graph because it is easy to spot cycles (at least small ones).

5.8 Since m+n-1=6 is even, if no redundant braces are added the second player will win the building version. First can force a redundant brace on her/his third move. Can Second force another redundant brace and always win?

5.10 The 4 by whatever staircase grid graph needs a path of length 7.

SECTION 6

Activity 14

i) $M = \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix}$ $M' = \begin{matrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{matrix}$ $MM' = \begin{matrix} 2 & 0 \\ 0 & 1 \end{matrix}$ $M'M = \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$

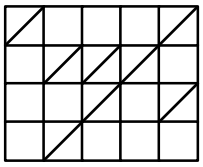
MM' and M'M are symmetric

ii) $A = \begin{matrix} 0&0&1&0&1\\ 0&0&0&1&0\\ 1&0&0&0&0\\ 0&1&0&0&0\\ 1&0&0&0&0 \end{matrix}$ $A+A^2+A^3+A^4 = \begin{matrix} 6&0&3&0&3\\ 0&2&0&2&0\\ 3&0&3&0&3\\ 0&2&0&2&0\\ 3&0&3&0&3 \end{matrix}$ 0's persist, thus the graph is not connected and the grid is not rigid

iii) $A = \begin{matrix} 0&0&1&0&1\\ 0&0&0&1&1\\ 1&0&0&0&0\\ 0&1&0&0&0\\ 1&1&0&0&0 \end{matrix}$ $A+A^2+A^3+A^4 = \begin{matrix} 7&5&3&1&4\\ 5&7&1&3&4\\ 3&1&3&1&4\\ 1&3&1&3&4\\ 4&4&4&4&8 \end{matrix}$ no 0's, thus the graph is connected and the grid is rigid

note the grid matrix M is in the top-right corner of A and M' is in bottom-left so A can be partitioned or blocked as $A = \begin{matrix} O_{2,2} & M_{2,3} \\ M'_{3,2} & O_{3,3} \end{matrix}$

Activity 15

1 i) The grid is 

$MM'M = \begin{matrix} 2&0&1&0&3\\ 0&4&4&3&1\\ 1&1&3&1&3\\ 0&2&1&1&0 \end{matrix}$ (some 0's so we can't conclude the grid is rigid yet)

$MM'MM'M = \begin{matrix} 5&1&5&1&9\\ 1&15&16&11&6\\ 4&6&11&5&10\\ 0&6&5&4&1 \end{matrix}$ (still a 0 in 4,1 postion - so there is no path of length 5 or less from row 4 to column 1 in the grid graph)

However the 7 product has all nonzeros: the graph is connected, the grid is rigid

2 $M = \begin{matrix} 1&1&1&1\\ 1&0&0&0\\ 1&0&0&0 \end{matrix}$ $Q = MM'MM'M = \begin{matrix} 30&18&18&18\\ 12&6&6&6\\ 12&6&6&6 \end{matrix}$ no 0's so gamma is rigid

$M = \begin{matrix} 1&0&1&0\\ 0&1&0&1\\ 1&0&1&0 \end{matrix}$ $Q = MM'MM'M = \begin{matrix} 16&0&16&0\\ 0&4&0&4\\ 16&0&16&0 \end{matrix}$ checkerboard is not rigid

Exercises 6

6.1 If row i of M has all 0's then so will row i of MX for any matrix X. So row i of the Q matrix will be all 0's. Similarly for column j of M, XM and Q. For OBIRAC the M matrix will have a 1 in some i,j position with 0's elsewhere in row i and column j. This will persist in the odd MM' products so Q will have 0's in row i and column j except in the i,j position.

APPENDIX II ENRICHMENT WITH CALCULATORS AND COMPUTERS

USING DERIVE OR A GRAPHING CALCULATOR

The software package Derive will do calculations with matrices. From the command menu select Declare, then Matrix, and then give it the dimensions and entries of either the A or the M matrix. It will happily (and correctly) calculate $A + A^2 + A^3$ or MM'MM'M if you ask it to. Two quick hints: the transpose of M, M^t, is written M` (the "back prime" is below the tilde, ~, on my keyboard), and here is an example of the syntax for entering matrices directly M := [[1,0],[1,1]].

Many graphing calculators have a similar ability to calculate powers and products of matrices.

USING MATLAB

Matlab is a powerful and extensive software package which will not only calculate matrix sums and powers, but also draw graphs. Here is a sample showing the commands to deal with the 2 by 2 grid shown and its adjacency matrix, A.

```
% Define the adjacency matrix
A = [0 0 1 0; 0 0 1 1; 1 1 0 0; 0 1 0 0]
% Say where to place the vertices
xy = [ 1 2; 1 1; 2 2; 2 1]
axis([0 3 0 3])
gplot(A, xy)
% Ask for the sum of powers of A
B = A + A*A + A*A*A
```

If you "know" Matlab you might try to write a program to allow the user to enter just the M matrix instead of the whole A matrix.

USING MATHEMATICA

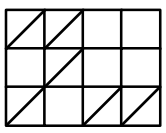
Both in version 2.2.3 and in 3.0, Mathematica comes with a package containing specific functions useful for graph theory. Open the package with the command **<<DiscreteMath`Combinatorica`**

All functions that apply to a graph need as input the adjacency matrix and the set of vertices. Since we have seen in section 6 that the adjacency matrix consists of the matrix M, its transpose and blocks of zero, you can use the following function to save you some work. The function will use as input the grid matrix, and its output will be the adjacency matrix.

```
BuildAdjacency[gm_List]:=   Module[
{m = Dimensions[gm][[1]],    n = Dimensions[gm][[2]],
 gmt = Transpose[gm]},       A = Table[0,{i,n+m},{j,n+m}];
Do[A[[i]] = Flatten[Prepend[gm[[i]],Table[0,{m}]]],{i,m}];
Do[A[[j]] = Flatten[Append[gmt[[j-m]],Table[0,{n}]]],{j,m+1,m+n}];
A]
```

For example, let M be the grid matrix for the 3 by 4 grid shown.

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$



Enter the matrix either as a list of rows, or in version 3.0, you can use the matrix template. Below is how you would enter it in version 2.2.3. If you call the function BuildAdjacency with the output form //MatrixForm, your output will be easy to read off. Since we want to use the adjacency matrix later, we will give it the name A:

```
gm = {{1,1,0,0},{0,1,0,0},{1,0,1,1}};

A = BuildAdjacency[gm]//MatrixForm
```

0	0	0	1	1	0	0
0	0	0	0	1	0	0
0	0	0	1	0	1	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	1	0	0	0	0

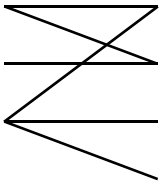
Now that we have created the adjacency matrix, we need to create the list of vertices. The following function will create the list of vertices using the matrix M, by reading off the number of rows and columns. The resulting list of vertices will be in the "standard" form we have been using for our graphs.

```
BuildVertexList[gm_List]:=
Module[
{m = Dimensions[gm][[1]],
 n = Dimensions[gm][[2]]},
 v = Join[Reverse[Table[{1,i},{i,n-m+1,n}]],
                  Reverse[Table[{2,j},{j,1,n}]]]]
```

Now we are ready to use some of the built-in functions. We can define a graph with the given adjacency matrix and the vertex set v using the function graph[A,v]. We can view the graph with the function ShowGraph[g].

```
v = BuildVertexList[gm];
g=Graph[A,v];
ShowGraph[g]
```

This gives the grid graph as shown.



We can find out if the graph is connected without actually drawing it by using the function ConnectedQ[g]. (For Q think "question"). The result is either True , i.e the graph is connected, or False.

```
ConnectedQ[g]
True
```

In our example, we already knew that the graph was connected, so the answer is not too surprising. We can ask the question how "well" connected the graph is, i.e. how many braces could be taken away at most before the graph gets disconnected. The function

EdgeConnectivity[g] computes the minimum number of edges whose deletion from graph g disconnects it.

```
EdgeConnectivity[g]
1
```

Thus this result tells us that if we take away one brace, the graph will become disconnected, i.e. the grid will be floppy. Again this is no surprise as we know from the m+n-1 principle that our grid is minimally braced - deleting any edge would make the grid floppy.

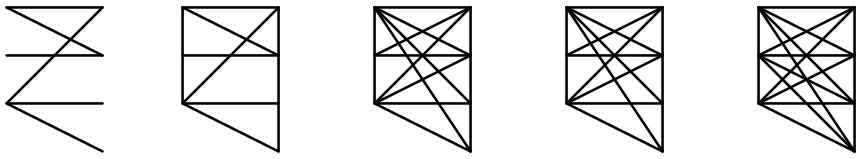
One of the results of Section 6 related to the adjacency matrix stated that the sum of powers of the adjacency matrix has to have all positive entries to show connectivity. Again we can use a built-in function, GraphPower[g,k], which computes the sum of powers less than or equal to k, to help us with computations. The output of this function will be the matrix sum, together with the list of the vertices. In this case it turns out that the matrix sum becomes positive starting with the fifth power.

```
GraphPower[g,5]
```

The output is actually in the form of a graph. This allows us to depict how paths of increasing length get created. In the graph created by GraphPower[g,k], an edge between v_i and v_j indicates that there is a path of length less than or equal to k between the two vertices.

```
s1 = ShowGraph[GraphPower[g,1]];
s2 = ShowGraph[GraphPower[g,2]];
s3 = ShowGraph[GraphPower[g,3]];
s4 = ShowGraph[GraphPower[g,4]];
s5 = ShowGraph[GraphPower[g,5]];

Show[GraphicsArray[{s1,s2,s3,s4,s5}]]
```



Dr. Constance C. Edwards
Mathematics Department
Coastal Carolina University
804 Berrywood Court
Myrtle Beach SC 29579

Dr. Silvia Heubach
Mathematics Department
California State University
5151 State University Drive
Los Angeles CA 90032-8204

Prof. Vernon Howe
Department of Math and Computing
La Sierra University
Riverside CA 92515

Prof. Gary Klatt
Mathematics and C.S. Department
Univ. of Wisconsin - Whitewater
Whitewater, WI 53190